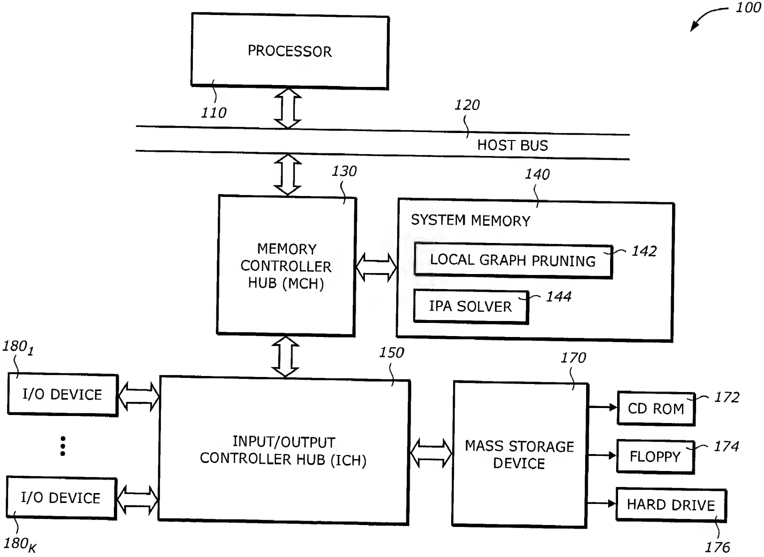
100
PROCESSOR
110
120
HOST BUS
130
MEMORY CONTROLLER HUB (MCH)
140
SYSTEM MEMORY
LOCAL GRAPH PRUNING
142
IPA SOLVER
144
180_1
I/O DEVICE
⋮
I/O DEVICE
180_K
150
INPUT/OUTPUT CONTROLLER HUB (ICH)
170
MASS STORAGE DEVICE
CD ROM
172
FLOPPY
174
HARD DRIVE
176

FIG. 1

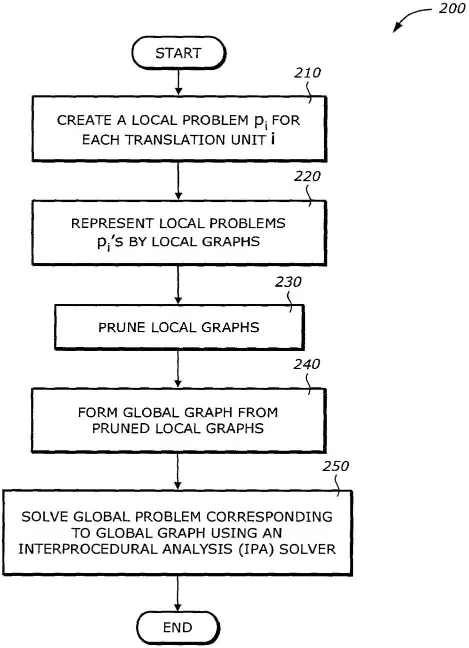
200
START
210
CREATE A LOCAL PROBLEM p_i FOR EACH TRANSLATION UNIT i
220
REPRESENT LOCAL PROBLEMS p_i'S BY LOCAL GRAPHS
230
PRUNE LOCAL GRAPHS
240
FORM GLOBAL GRAPH FROM PRUNED LOCAL GRAPHS
250
SOLVE GLOBAL PROBLEM CORRESPONDING TO GLOBAL GRAPH USING AN INTERPROCEDURAL ANALYSIS (IPA) SOLVER
END

FIG. 2

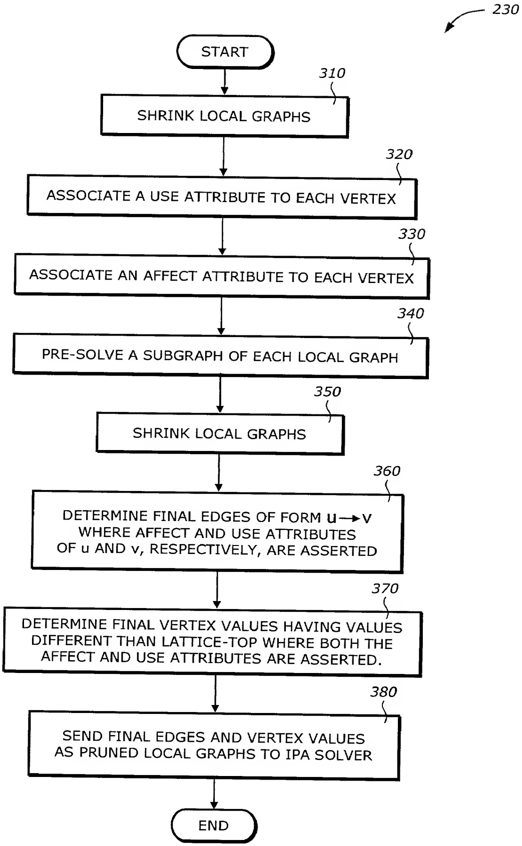
230
START
310
SHRINK LOCAL GRAPHS
320
ASSOCIATE A USE ATTRIBUTE TO EACH VERTEX
330
ASSOCIATE AN AFFECT ATTRIBUTE TO EACH VERTEX
340
PRE-SOLVE A SUBGRAPH OF EACH LOCAL GRAPH
350
SHRINK LOCAL GRAPHS
360
DETERMINE FINAL EDGES OF FORM u→v WHERE AFFECT AND USE ATTRIBUTES OF u AND v, RESPECTIVELY, ARE ASSERTED
370
DETERMINE FINAL VERTEX VALUES HAVING VALUES DIFFERENT THAN LATTICE-TOP WHERE BOTH THE AFFECT AND USE ATTRIBUTES ARE ASSERTED.
380
SEND FINAL EDGES AND VERTEX VALUES AS PRUNED LOCAL GRAPHS TO IPA SOLVER
END

FIG. 3

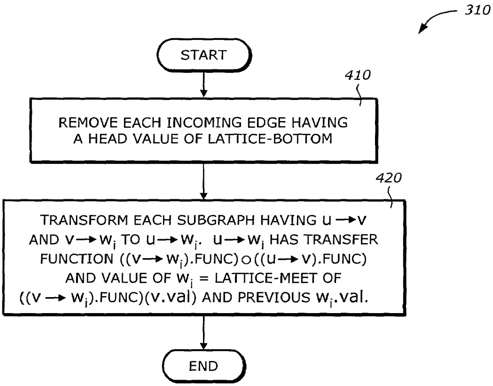
310
START
410
REMOVE EACH INCOMING EDGE HAVING A HEAD VALUE OF LATTICE-BOTTOM
420
TRANSFORM EACH SUBGRAPH HAVING $u \rightarrow v$ AND $v \rightarrow w_i$ TO $u \rightarrow w_i$. $u \rightarrow w_i$ HAS TRANSFER FUNCTION $((v \rightarrow w_i).\mathrm{FUNC}) \circ ((u \rightarrow v).\mathrm{FUNC})$ AND VALUE OF w_i = LATTICE-MEET OF $((v \rightarrow w_i).\mathrm{FUNC})(v.\mathrm{val})$ AND PREVIOUS $w_i.\mathrm{val}$.
END

FIG. 4A

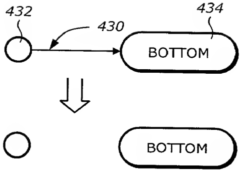
432
434
430
BOTTOM
BOTTOM

FIG. 4B

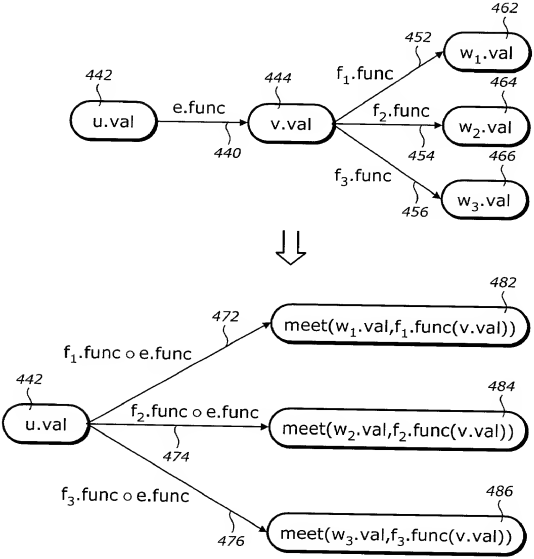
462
452
442
444
f1.func
w1.val
464
u.val
e.func
v.val
f2.func
w2.val
440
454
466
f3.func
w3.val
456
482
472
meet(w1.val,f1.func(v.val))
f1.func o e.func
442
484
u.val
f2.func o e.func
meet(w2.val,f2.func(v.val))
474
486
f3.func o e.func
meet(w3.val,f3.func(v.val))
476

FIG. 4C

320

```
Procedure ASSOCIATE_USE_ATTRIBUTE(g)
g:graph

begin
    for each vertex u in g, do
        u.uses_named_vertex:= false;
    enddo
    for each vertex u do
        if u is a named vertex then
            MARK_USES(u)
        endif
    enddo
end

Procedure MARK_USES(u)
u:vertex;

begin
    if (not u.uses_named_vertex) then
        u.uses_named_vertex:= true;
        for each edge of form u → v do
            MARK_USES(v)
        enddo
    endif
end
```

FIG. 5

330

```
Procedure ASSOCIATE_AFFECTS_ATTRIBUTE(g)
g:graph

begin
    for each vertex v in g, do
        v.affects_named_vertex:= false;
    enddo
    for each vertex v do
        if v is a named vertex then
            MARK_AFFECTS(v)
        endif
    enddo
end

Procedure MARK_USES(v)
v:vertex;

begin
    if (not v.affects_named_vertex) then
        v.affects_named_vertex:= true;
        for each edge of form u→v do
            MARK_AFFECTS(u)
        enddo
    endif
end
```

FIG. 6

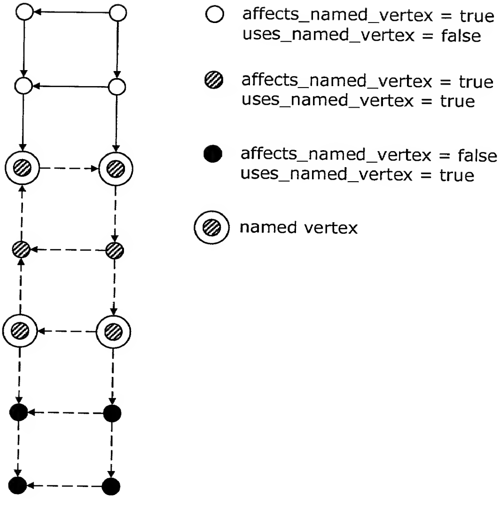
○ affects_named_vertex = true
uses_named_vertex = false
▨ affects_named_vertex = true
uses_named_vertex = true
● affects_named_vertex = false
uses_named_vertex = true
named vertex

FIG. 7

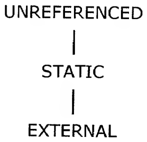

FIG. 8

```
/*TRANSLATION UNIT #1*/

extern void c();
extern void e();

static void f(){
    c();
    e();
}
void d(){
    f();
}
void e(){
}
main(){
    d();
}
```

```
/*TRANSLATION UNIT #2*/

void b();
static void (*a)()=b;

extern void d();
extern void e();

void b(){
    d();
    (*a)();
}
void c(){
    e();
}
```

FIG. 9

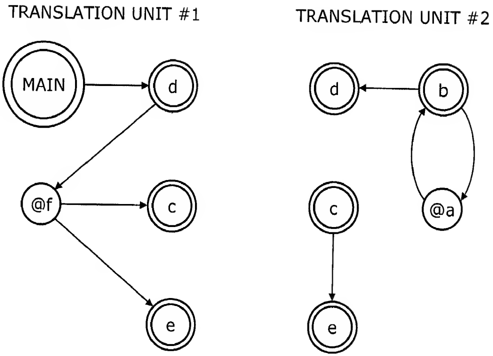

FIG. 10

UNIT #1 EDGES	UNIT #2 EDGES	FUNCTION
main → d d → @f @f → c	b → @a @a → b	{S,E} → S; U → U
@f → c	c → e b → d	{S,E} → E; U → U

FIG. 11

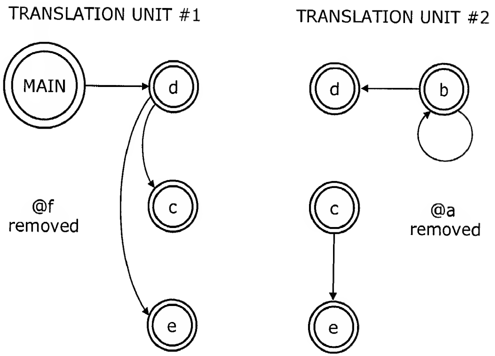
TRANSLATION UNIT #1
MAIN
d
c
e
@f
removed
TRANSLATION UNIT #2
d
b
c
e
@a
removed

FIG. 12

UNIT #1 EDGES	UNIT #2 EDGES	FUNCTION
main → d d → e	b → b	{S,E} → S; U → U
d → c	c → e b → d	{S,E} → E; U → U

FIG. 13

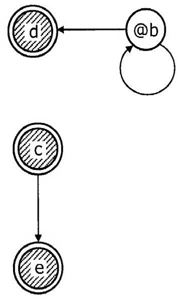

FIG. 14

340

```
Procedure PRESOLVE(g)
g:graph

begin
    do
        changed = false;
        for each edge u→v in g do
            if (not u.uses_named_vertex) then
                t := meet(v.val,((u->v).func)(u.val));
                if t ≠ v.val then
                    v.val := t;
                    changed := true;
                endif
            endif
        enddo
    while changed;
end
```

FIG. 15